# Ejercicios de Repaso de Clases (R06)

Sacado del Examen del 2º trimestre del curso 2014-2015

Escribe las clases **PokerCard**, **PokerHand**, **PokerDeck** que nos permitirán escribir un programa de VideoPoker.

- Clase **PokerCard**: representará una carta de póker.
    - Atributos: **rank**, que representará el número, y **suit**, que representará el palo. Cada uno usará un *enum* con los valores, que os pongo a continuación:

```
enum Suit { Hearts, Diamonds, Spades, Clubs };
enum Rank { N2, N3, N4, N5, N6, N7, N8, N9, N10, Jack, Queen, King, Ace };
```

    - Constructor: un constructor al que le pasamos el número y el palo y los guarda en los atributos.
    - Propiedades: getSuit() y getRank() para obtener el número y el palo de la carta.
    - Métodos:
        - toString() que nos mostrará la carta con su número y su simbolito para el palo. Ej: A♠ 10♦

- Clase **PokerHand**: representará una mano (cinco cartas) de póker.
    - Atributos: **hand**, un array de cinco cartas de póker (**PokerCard**).
    - Constructor: un constructor con cinco parámetros (cinco PokerCard). El constructor meterá las cinco cartas en el array y luego las ordenará con el método *sortHand*.
    - Métodos:
        - toString(), nos mostrará las cinco cartas por pantalla en una sola línea.
        - sortHand(), nos ordenará la mano, lo cual hará que se pueda ver mejor si hay alguna combinación interesante. Las cartas estarán ordenadas primero por número y luego por palo.
          Ej.: [A♥, 10♠, 5♠, 10♥, 5♣] => [5♣, 5♠, 10♥, 10♠, A♥]

        - *boolean* isPair(), nos devolverá *true* si hay al menos una pareja, es decir, si hay al menos dos cartas con el mismo número.

        - *boolean* isTwoPairs(), nos devolverá *true* si hay al menos dos parejas.

        - *boolean* isThree(), nos devolverá *true* si hay al menos un trio.

        - *boolean* isStraight(), nos devolverá *true* si hay escalera (los números de las cinco cartas son consecutivos).

        - *boolean* isFlush(), nos devolverá *true* si hay color (todas las cartas son del mismo palo).

        - *boolean* isFull(), nos devolverá *true* si hay full (un trío y una pareja).

        - *boolean* isPoker(), nos devolverá *true* si hay póker (cuatro cartas iguales).

        - *boolean* isStraightFlush(), nos devolverá *true* si hay escalera de color (escalera + color, podéis usar las otras dos funciones).

        - *b*oolean isRoyalFlush(), nos devolverá *true* si hay una escalera de color que acaba en un as (podéis usar la función anterior y comprobar si la última carta es un as).

- Clase **PokerDeck**: representará una baraja de póker.

    - Atributos: **deck**, una lista de PokerCard.

    - Constructor: un constructor sin parámetros, que creará la lista e introducirá en ella las 52 cartas de la baraja de póker. A continuación, barajará la baraja.

    - Métodos:
        - *PokerCard* drawCard(), robará una carta de la baraja y nos la devolverá.

Una vez tengamos estas tres clases terminadas y probadas, podremos implementar el juego del Video Póker. El juego funciona de la siguiente manera:

- Se roban 5 cartas que se muestran al jugador por pantalla.
- Se le da la posibilidad al jugador de descartarse de las cartas que no le interesen.
- Se le dan cartas a cambio de las descartadas.
- Dependiendo de la mano que obtenga al final, se le da un premio. Si no tiene ni siquiera pareja, pierde su dinero. Si tiene al menos una pareja, se le da un premio dependiendo de lo buena que sea la mano:

| **Hand** | **Prize** |
|---|---|
| Royal Flush | 800 |
| Straight Flush | 50 |
| Four of a kind | 25 |
| Full House | 9 |
| Flush | 6 |
| Straight | 4 |
| Three of a kind | 3 |
| Two Pair | 2 |
| Jacks or Better | 1 |

Estos premios se deben comprobar de mayor a menor y se le da el premio de lo mayor que tenga.

Para hacer el juego más completo, el jugador debería empezar con un saldo y después de cada partida se le dice el saldo actual. El juego termina si se queda sin dinero.